

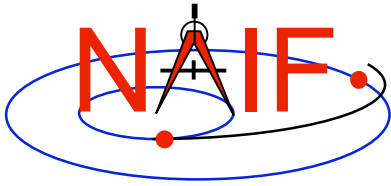


Navigation and Ancillary Information Facility

MATLAB Interface to CSPICE “Mice”

How to Access the CSPICE library Using MATLAB[©]

January 2008



Topics

Navigation and Ancillary Information Facility

- **Mice Benefits**
- **How does it work?**
- **Distribution**
- **Mice Operation**
- **Vectorization**
- **Simple Use of Mice Functionality**



Mice Benefits

Navigation and Ancillary Information Facility

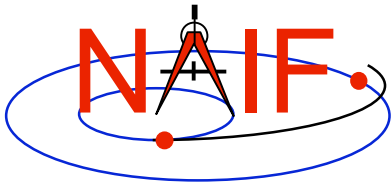
- Mice operates as an extension to the MATLAB language regime.
- All Mice calls are functions regardless of the call format of the underlying CSPICE routine, returning MATLAB native data types.
- Mice has some capability not available in CSPICE such as vectorization.
- CSPICE error messages return to MATLAB in the form usable by the *try...catch* construct.



How Does It Work? (1)

Navigation and Ancillary Information Facility

- The MATLAB environment includes an intrinsic capability to use external routines.
 - Mice functions as a MATLAB Executable, MEX, consisting of the Mice MEX shared object library and a set of .m wrapper files.
 - » The Mice library contains the MATLAB callable C interface routines that wrap a subset of CSPICE wrapper calls.
 - » The wrapper files, named `cspice_*.m` and `mice_*.m`, provide the MATLAB calls to the interface functions.
 - » A function prefixed with 'cspice_' retains essentially the same argument list as the CSPICE counterpart.
 - » An interface prefixed with 'mice_' returns a structure, with the fields of the structure corresponding to the output arguments of the CSPICE counterpart.
 - » The wrappers include a header section describing the function call, displayable by the MATLAB *help* command.



How Does It Work? (2)

Navigation and Ancillary Information Facility

When a user invokes a call to a Mice function:

- 1. MATLAB calls...**
- 2. the function's wrapper, which calls...**
- 3. the Mice MEX shared object library, which performs its function then returns the result...**
- 4. to the wrapper, which...**
- 5. returns the result to the user**

... transparent from the user's perspective.



Mice Distribution

Navigation and Ancillary Information Facility

- **NAIF distributes Mice as a complete, standalone package.**
- **The package includes:**
 - The CSPICE source files.
 - The Mice interface source code.
 - Platform specific build scripts for Mice and CSPICE.
 - MATLAB versions of the SPICE cookbook programs, *states*, *tictoc*, *subpt*, and *simple*.
 - An HTML based help system for both Mice and CSPICE, with the Mice help cross-linked to CSPICE.
 - The Mice MEX shared library and the M wrapper files. The system is ready for use after installation of the the library and wrapper files.



Mice Operation (1)

Navigation and Ancillary Information Facility

- **A possible irritant exists in loading kernels using the `cspice_furnsh` function.**
 - Kernels load into your MATLAB session, not into your MATLAB scripts. This means:
 - » loaded binary kernels remain accessible (“active”) throughout your MATLAB session
 - » data from loaded text kernels remain in the kernel pool (in the memory space used by CSPICE) throughout your MATLAB session
 - Consequence: some kernel data may be available to one of your scripts even though not intended to be so.
 - » You could get **incorrect results!**
 - » (If you run only one script during your MATLAB session, there’s no problem.)

continued on next page



Mice Operation (2)

Navigation and Ancillary Information Facility

- **Mitigation: two approaches**
 - Load all needed SPICE kernels for your MATLAB session at the beginning of the session, paying careful attention to the files loaded and the loading order (loading order affects precedence)
 - » Convince yourself that this approach will provide **ALL** of the scripts you will run during this MATLAB session with the appropriate SPICE data
 - At or near the end of every MATLAB script:
 - » include a call to `cspice_unload` for each kernel loaded using `cspice_furnsh`
 - » or include a call to `cspice_kclear` to remove **ALL** kernel data from the kernel pool loaded using `cspice_furnsh`



Mice Vectorization (1)

Navigation and Ancillary Information Facility

- **Most Mice functions include use of vectorized arguments, a capability not available in C or FORTRAN toolkits.**
- **Example: use Mice to retrieve state vectors and light-time values for 1000 ephemeris times.**
 - **Create the array of 1000 ephemeris times in steps of 10 hours, keyed on July 1, 2005:**

```
start = cspice_str2et('July 1 2005');  
et     = (0:999)*36000 + start;
```

- **Retrieve the state vectors and corresponding light times from Mars to earth at each `et` in the J2000 frame with LT+S aberration correction:**

```
[state, ltime] = cspice_spkezr( 'Earth', et, 'J2000', 'LT+S', 'MARS');
```

continued on next page



Mice Vectorization (2)

Navigation and Ancillary Information Facility

- Access the *ith* state 6-vector (6x1 array) corresponding to the *ith* ephemeris time with the expression

```
state_i = state(:,i)
```

- Convert the ephemeris time vector `et` from the previous example to UTC calendar strings with three decimal places of precision in the seconds field.

```
format = 'C';  
prec   = 3;  
utcstr = cspice_et2utc( et, format, prec );
```

- The call returns `utcstr`, an array of 1000 strings (dimensioned 1000x24), where each *ith* string is the calendar date corresponding to `et(i)`. Access the *ith* string of `utcstr` corresponding to the *ith* ephemeris time with the expression

```
utcstr_i = utcstr(i,:)
```



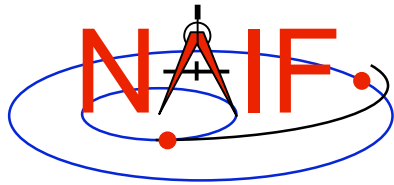
Mice Vectorization (3)

Navigation and Ancillary Information Facility

- **Convert the position components (the first three components in a state vector) of the N state vectors returned in `state` to latitudinal coordinates.**

```
[radius, latitude, longitude] = cspice_reclat( state(1:3,:) );
```

- **The call returns three double precision 1x1000 arrays (vectorized scalars): `radius`, `latitude`, and `longitude`.**



Simple Use of Mice Functionality

Navigation and Ancillary Information Facility

- **As an example of Mice use, calculate and plot the trajectory in the J2000 inertial frame of the Cassini spacecraft from June 20, 2004 to December 1, 2005.**

```
% Define the number of divisions of the time interval and the time interval.
STEP = 1000;
utc  = strvcat( 'Jun 20, 2004', 'Dec 1, 2005' )

% Load the needed kernels. Use a meta kernel "standard.ker" to load the kernels
% "naif0008.tls," "de405_2000-2050.bsp," "pck00008.tpc."
cspice_furnsh( 'standard.ker' )
cspice_furnsh( '/kernels/cassini/spk/T18-5TDJ5.bsp' )

et      = cspice_str2et( utc );
times   = (0:STEP-1) * ( et(2) - et(1) )/STEP + et(1);
[pos,ltime ]=cspice_spkpos( 'Cassini', times, 'J2000', 'NONE', 'SATURN BARYCENTER' );

% Plot the resulting trajectory.
x = pos(1,:);
y = pos(2,:);
z = pos(3,:);

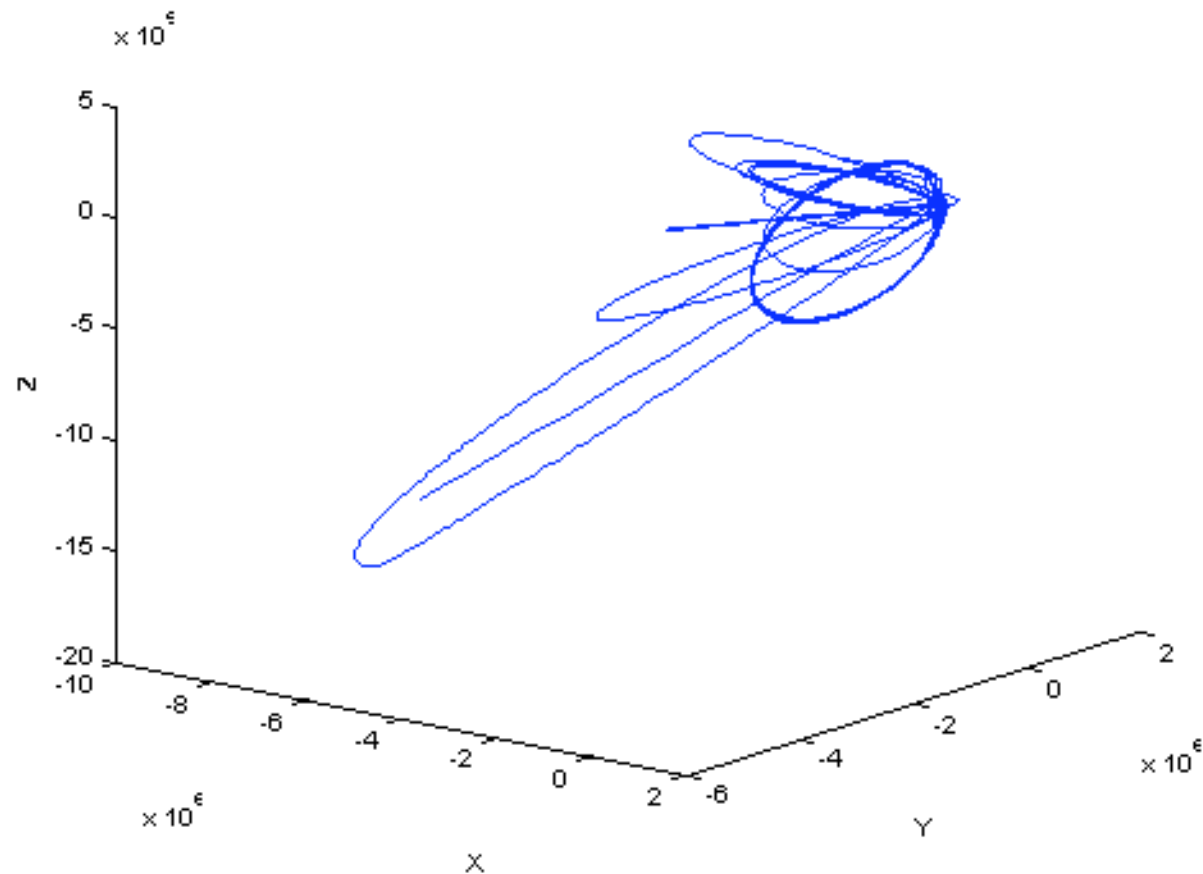
plot3(x,y,z)

cspice_kclear
```



Graphic Output using MATLAB

Navigation and Ancillary Information Facility



Trajectory of the Cassini vehicle in the J2000 frame, for June 20, 2005 to Dec 1, 2005